Rules, RIF and RuleML

Rule Knowledge

- Rules generalize facts by making them conditional on other facts (often via chaining through further rules)
- Rules generalize taxonomies via multiple premises, n-ary predicates, structured arguments, etc.
- Two uses of rules - *top-down* (backward-chaining) and *bottom-up* (forward-chaining) - represented only once
- To avoid n^2–n *pairwise* translators: Int'l standards with 2n–2 *in-and-out* translators:
  - RuleML: Rule Markup Language (work with ISO, OMG, W3C, OASIS)
    - Deliberation RuleML 1.0 released as a de facto standard
  - ISO: Common Logic (incl. CGs & KIF: Knowledge Interchange Format)
    - Collaboration on Relax NG schemas for XCL 2 / CL RuleML
  - OMG: Production Rules Representation (PRR), SBVR, and API4KB
  - W3C: Rule Interchange Format (RIF)
    - Gave rise to open-source and commercial RIF implementations
  - OASIS: LegalRuleML

The interchange approach

- W3C's RDF stack is an integrated solution for encoding & interchanging knowledge
  - Supporting OWL (DL) constrains it quite a bit
  - E.g., preventing adoption of an OWL rule standard
- There are other approaches to standardiz-ing rule languages for knowledge exchange
  - RuleML: Rule Markup Language, an XML approach for representing rules
  - RIF: Rule Interchange Format, a W3C standard for exchanging rules
- Neither tries to be compatible with OWL

Many different rule languages

- There are rule languages families: logic, logic programming, production, procedural, etc.
  - Instances in a family may differ in their syntax, semantics or other aspects
- Jess production rule language

  (defrule r42 (parent ?a ?b) (male ?a)
  => (assert (father ?a ?b)))
- Prolog logic programming language

  father(A,B) :- parent(A,B), Male (A).
- Common Logic logic format

  (=> (and (paent ?a ?b) (male ?a)) (father ?a ?b))

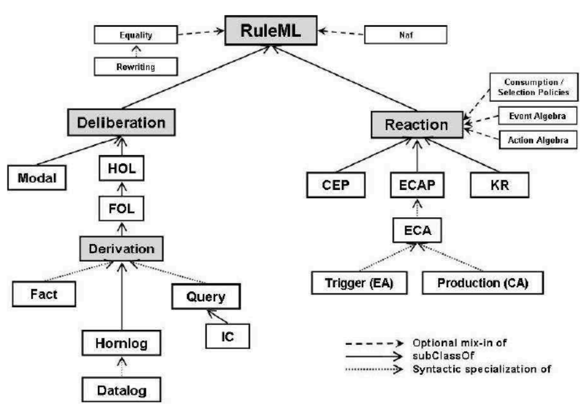
X Interchange Format

- Rather than have N^2 translators for N languages, we could
  - Develop a common rule interchange format
  - Let each language do import/export mappings for it
- Two modern interchange formats for rules
  - RuleML: Rule Markup Language, an XML approach for representing rules
  - RIF: Rule Interchange Format, a W3C standard for exchanging rules

RuleML

- RuleML's goal: express both forward (bottom-up) and backward (top-down) rules in XML
- See http://ruleml.org/
- Effort began in 2001 and has informed and been informed by W3C efforts
- An "open network of individuals and groups from both industry and academia"

Taxonomy of RuleML rules



from Boley et. al., RuleML 1.0: The Overarching Specification of Web Rules, 2010. http://bit.ly/RuleML

RIF

- W3C Rule Interchange Format
- Three dialects: Core, BLD, and PRD
  - Core: common subset of most rule engines, a "safe" positive datalog with builtins
  - BLD (Basic Logic Dialect): adds logic functions, equality and named arguments, ~positive horn logic
  - PRD (Production Rules Dialect): adds action with side effects in rule conclusion
- Has a mapping to RDF

An example of a RIF rule

From http://w3.org/2005/rules/wiki/Primer

```
Document(
 Prefix(rdfs <http://www.w3.org/2000/01/rdf-schema#>)
 Prefix(imdbrel <http://example.com/imdbrelations#>)
 Prefix(dbpedia <http://dbpedia.org/ontology/>)
 Group(
  Forall ?Actor ?Film ?Role (
   If And(imdbrel:playsRole(?Actor ?Role)
          imdbrel:roleInFilm(?Role ?Film))
   Then dbpedia:starring(?Film ?Actor) ) ) )
```

Another RIF example, with guards

From http://w3.org/2005/rules/wiki/Primer

```
Document(
 Prefix(rdf <http://www.w3.org/1999/02/22-rdf-syntax-ns#>)
 Prefix(rdfs <http://www.w3.org/2000/01/rdf-schema#>)
 Prefix(imdbrel <http://example.com/imdbrelations#>)
 Prefix(dbpedia http://dbpedia.org/ontology/)
 Group(
  Forall ?Actor ?Film ?Role (
   If   And(?Actor # imdbrel:Actor
            ?Film # imdbrel:Film
            ?Role # imdbrel:Character
            imdbrel:playsRole(?Actor ?Role)
            imdbrel:roleInFilm(?Role ?Film))
   Then dbpedia:starring(?Film ?Actor) )))
```

Rif document can contain facts

The following will conclude *bio:mortal(phil:Socrates)*

```
Document(
  Prefix(bio <http://example.com/biology#>)
  Prefix(phil <http://example.com/philosophers#>)
  Group(
   If  bio:human(?x)
   Then bio:mortal(?x) )
  Group(
   bio:human(phil:Socrates) ))
```

Another RIF example (PRD)

From http://w3.org/2005/rules/wiki/Primer

```
Document(
  Prefix(rdfs <http://www.w3.org/2000/01/rdf-schema#>)
  Prefix(imdbrelf <http://example.com/fauximdbrelations#>)
  Prefix(dbpediaf <http://example.com/fauxibdbrelations>)
  Prefix(ibdbrelf <http://example.com/fauxibdbrelations#>)
  Group(
   Forall ?Actor (
    If   Or(Exists ?Film (imdbrelf:winAward(?Actor ?Film))
            Exists ?Play (ibdbrelf:winAward(?Actor ?Play)) )
    Then assert(dbpediaf:awardWinner(?Actor)) )
  imdbrelf:winAward(RobertoBenigni LifeIsBeautiful) ))
```

Why do we need YAKL

- YAKL: Yet another knowledge language
- Rules are good for representing knowledge
- Rule idioms have powerful features that are not and can not be supported by OWL
  - Non-monotonic rules
  - Default reasoning
  - Arbitrary functions, including some with with side effects
  - etc.

Non-monotonic rules

- Non-monotonic rules use an “unprovable” operator
- This can be used to implement default reasoning, e.g.,
  - assume P(X) is true for some X unless you can prove hat it is not
  - Assume that a bird can fly unless you know it can not

monotonic

```
canFly(X) :- bird (X)
bird(X) :- eagle(X)
bird(X) :- penguin(X)
eagle(sam)
penguin(tux)
```

Non-monotonic

```
canFly(X) :- bird (X), \+ not(canFly(X))
bird(X) :- eagle(X)
bird(X) :- penguin(X)
not(canFly(X)) :- penguin(X)
eagle(sam)
penguin(tux)
```

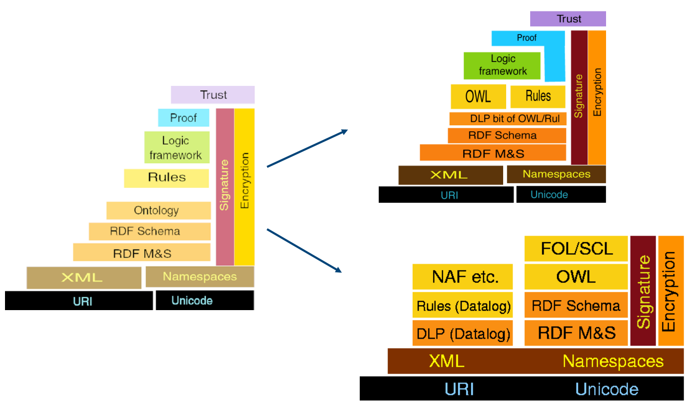
Default rules in Prolog

- In prolog it's easy to have
  - Default(?head :- ?body).
- Expand to
  - ?head :- ?body, +\ not(?head) .
- So
  - default(canFly(X) :- bird(X))
- Expands to
  - canFly(X) :- bird(X), \+(not(canFly(X))).

Rule priorities

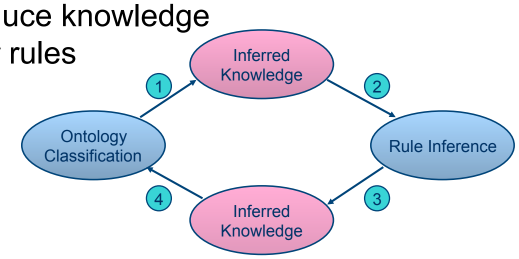
- This approach can be extended to implement systems where rules have priorities
- This seems to be intuitive to people – used in many human systems
  - E.g., University policy overrules Department policy
  - The "Ten Commandments" can not be contravened

Two Semantic Webs?



Limitations

- The rule inference support not integrated with OWL classifier
  - New assertions by rules may violate existing restrictions in ontology
  - New inferred knowledge from classification may produce knowledge useful for rules



Limitations

- Existing solution: solve possible conflicts manually
- Ideal solution: a single module for both ontology classification and rule inference
- What if we want to combine non-monotonic features with classical logic?
- Partial Solutions:
  - Answer set programming
  - Externally via appropriate rule engines

Summary

- Horn logic is a subset of predicate logic that allows efficient reasoning, orthogonal to description logics
- Horn logic is the basis of monotonic rules
- DLP and SWRL are two important ways of combining OWL with Horn rules.
  - DLP is essentially the intersection of OWL and Horn logic
  - SWRL is a much richer language

Summary (2)

- Nonmonotonic rules are useful in situations where the available information is incomplete
- They are rules that may be overridden by contrary evidence
- Priorities are sometimes used to resolve some conflicts between rules
- Representation XML-like languages is straightforward